To share or not to share: code sharing in social science

The below text is a transcript of a session that was held on Monday, 25 October 2021 at the 2021 Research Methods e-Festival.

The title of the session was “To share or not to share: code sharing in social science”. The speakers were Louise Corti of the Office for National Statistics, Professor Felix Ritchie of the University of West England, and Martin O’Reilly of the Alan Turing Institute.

Louise Corti (LC) and Felix Ritchie (FR) interview

LC: Can you tell us a little about yourself and your work?

FR: I am a microeconomist and have been for the past 30 years, I typically analyse ‘record-level data’ (individual responses, for example from surveys or administrative data) using a statistical program and publish my research findings. Prior to that I was a programmer working for a company writing office management software.

LC: What experience have you had with writing, sharing or checking code. Have there been any challenges?

FR: So, from a very early point in my working life age, I was trained to write code so that other people could understand it - including those in your own team! Imposing standards means, for example, that that you are consistent in the way you use variable names and lay out your code. While working as a researcher in social science research is usually a solitary occupation, I still tend to follow standards so that I can return to my own work later and remember what I've done.

At the moment I'm collaborating much more with colleagues. I am leading a large project where eight of us are bringing together multiple surveys including the Census from ONS and information from the tax department. During our quality assessment and data preparation work we are trying to create new data assets which other researchers will be able to use for their analyses.

So, it’s really important that we document data and code as we go along, for example with variables explained, so that code from the original data used to generate to clean dataset can be rerun, consulted and checked. Even within a small group of us this is quite tricky because everyone has come from different traditions and backgrounds, and has their own way of doing things. Our solution has been to enforce an agreed higher-level approach, whilst letting individuals use their own styles of writing and layout, and learn how others are working.

LC: In your own discipline, is there a demand for being reproducible as a researcher? Are there any standards you are asked to meet?

FR: In my own discipline of economics, there is no single standard. Some journals will simple take the article and evaluate the paper on the basis of what you've submitted. Others will ask for more detailed information about the data, or ask for more descriptive tables. Some, like the American Economic Association, have come to what we might consider to be the gold standard, as they require you to provide the data and the code, and pass these through a validation process, operated via the journal’s data editor. This extreme end of the reproducibility spectrum is a very expensive option.

LC: Are there any particular challenges with doing research within secure environments?

FR: When a journal asks you to submit data to back up your analytic findings, and you have been working in a secure environment, you obviously can't release the data. Thus it's harder for resulting analyses to be directly reproducible.

However, one way to make our work more reproducible has been to share the code, rather than the data. Any code taken out of the TRE still needs to be reviewed to make sure that there is no disclosive information contained, such as results noted in the comments.

Documenting code should explain to others what you have done and how and why. If you simply use menus to analyse data, and don't examine the code behind it then it' going to be really difficult for anyone else to check your work and be confident that what you've done is what you've said you've done.

In France, *cascad* operates a service where you can pay to have experts go through and verify that your code does is actually what it says it did, and gain a Reproducibility certification.

LC: Are there any issues around who owns code and its reliability or the need for disclaimers from creators or data owners when others use it?

FR: I do believe that the Intellectual Property (IP) for code should stay with the researcher, given that we have put in so much effort. A good example of getting people to recognise and use your code was the GAUSS program DPD created by Manuel Arrellano & Stephen Bond, which was code for dynamic panel data models. Each time people used the data they were careful to reference the creators, recognizing how much effort had been put in. In my experience doing projects for Eurostat, where the code has been owned by the European Commission code, it is now published on GitHub under a Creative Commons license.

LC: Of course, alongside GitHub, the code could also be published with a formal reference and even a DOI (persistent identifier for a web-based research resource), such as that published in the data repository ReShare or Zenodo.

FR: While I haven't encountered any issues from data owners around publishing code, I am mostly concerned with ensuring what my team has done in its data work is clear and transparent. We would hope that data owners might be interested in value-added enhancements made to their data, and even QA it and include it in future releases. A collaborative approach is often beneficial.

LC: If we imagine the future 10 years from now, from a reproducibility perspective, what would you like to see in place?

FR: Given the fact that I work in a secure environment. I would like to see greater accessibility of code, including a guide to how it could be used. I also see opportunities for showing researchers how to make their code readable to and understandable by others, while also adopting an efficient approach.

I'd like to see coding viewed as a core part of the research process, and a valued investment, rather than being something that is just nice to have. My own approach is wanting to show what I have done. This is likely to suit the way our discipline (economics) is going, where it's becoming harder to get your stuff published if you don't have the code available.

LC: And, there are software tools available that can help you document, track and rerun your code. So, I think that it's all going in a positive direction.

Louise Corti (LC) and Martin O'Reilly (MOR) interview

LC: Can you tell us a little about yourself and your work?

MOR: I run a team of software engineers and data scientists at the Turing Institute, the UK's national institute for data science and AI, which is based in the British Library. We collaborate with and support our community of researchers to apply their research and make it more reproducible and reusable. Part of my own work is to guide them to work effectively and safely with sensitive data.

LC: What are the benefits of having research work be reproducible? What are best practices around code sharing in your area of work?

MOR: When we think of *reproducibility* we should first think of the benefits it brings for research. It allows others to scrutinize and validate our work, and is really a key part of the research process. This is simply impossible to do effectively without access to code and data.

Code sharing also enhances reusability; others can perform the same analysis on a different data set (replicablility) or can build on top of the code to extend their own methods. In today's world, so much research relies on complex data preparation workflows and computationally supported analyses, and it's just not possible to build on or extend work with only the information contained in the brief methods' section of a traditional research paper.

For the computational elements, of interest to the Turing Institute, the 'scholarship' comprises the data, the processing, the code and the computing environment used to generate the insights and results. These days some of this information can sit online in open lab books. This is useful for the community but also for ourselves, especially six months on a piece of analysis; your past self doesn't answer emails!

Reproducibility is a key tool to help us ensure our results are accurate, reliable and robust and it lets us move faster. We can have stronger confidence in our results as we carry out our research, able to compare experiments and knowing exactly what changed from one run to the next.

And it also lets us produce an easy, understandable presentation of our analysis - useful for looking at our own results as we change things, to communicate with our collaborators and for writing the final paper.

Good examples of code sharing include using collaborative version control, source code sharing systems, like GitHub and GitLab, which are available to all of us, and are increasingly being used. I'd encourage people to use these platforms from the very start, as they offer a great way to both share your code and track changes to your code, and promote collaboration.

To make the practice gold standard:

- the repository can be made publicly accessible
- a license can be added, thereby granting rights to others to use the materials. The Turing favours permissive open source licenses such as MIT or BSD, similar to CC-BY from Creative Commons, so that any reuse is permitted, even commercially, and all that is required is to acknowledge the contributor. Restrictions can be added in cases where intellectual property is a concern, or permitting use only where the user commits to sharing, such as the GPL
- a code repository can be linked with an archival repository (like Zenodo) so that the code gains a DOI, updates to code are reflected and it can be reliably cited in the longer term.
- core analysis methods can be packaged as reusable software packages and published in the standard repository for respective programming ecosystems, e.g. CRAN for R and PyPI for Python
- Write code in a way that's understandable to others and document it well. Literate programming environments can be used, such as Jupyiter Notebook and R Markdown, where code, analysis and results are interleaved together.

Regarding publishers in the computational and data science fields, it's still the exception rather than the norm to have research code required and validated when an article is submitted. However, publishers and conferences are increasingly requesting for submission both code and code availability statements.

Some publishers are trying out ways to allow readers to locate code more easily. ELife has established the Executable Research Article, which allows you to embed the code that produced your figures and your tables inside an open access article. Taylor Francis and Elsevier are piloting a similar commercial option for about a dozen journals each. Open source Binderhub offers one click from a GitHub repository containing your code to an R Studio or Jupyter Lab Notebook that reproduces your analysis. It captures the particular version of your code, and the software packages and operating system used. CODECHECK, by Steven Egland and colleagues, is a tool and service that can do gold standard checks on code and execute it to reproduce outcomes. It offers a 'certificate of executable computation', a bit like the cascad service. But it requires good code to make this process efficient!

LC: How can a researcher get started with being reproducible?

MOR: I think the big message is that you just start somewhere and each small step is better than not doing it; and you can build up from there.

Data provenance is really overlooked, so making available code that cleans data together with the final cleaned dataset can be useful, as Felix is doing in his project. Anything you can do to document the methodology used to create new data is beneficial (and also to yourself!) as well as automating processes where you can! Keep a detailed step-by-step account of your analysis and combine this with version control using GitHub or GitLab.

It can be daunting to share your code. I've been writing code professionally in industry and academia for almost 20 years, and I still write messy code when I'm working things out! The message I'd give to you is that your code probably isn't that bad! Sharing it openly helps you make it better.

You could start by making your analysis open at the end of project, simply to recreate tables in your paper, and you'll have achieved something. Then you can begin to document code as you go along. Using a reproducible computational environment could be a next step, so that your code and analysis are interleaved, for example using R or Python.

LC: So the combination of being brave, taking some first steps and using some of the great tools out there is a good start. And, connecting with Felix's message, working with colleagues on a project might help you to form some consensus on how to write shared code.

That leads us on to the 'Turing Way', can you tell us a bit about your handbook for reproducible data science?

MOR: The Turing Way is a lightly opinionated handbook for reproducible, ethical, and collaborative data science. It is the brainchild of Dr. Kirstie Whitaker, our Program Director for Tools, Practices and Systems at the Turing Institute seeking to build open research infrastructure that's accessible to the community.

The handbook curates and connects a range of resources and good practices with the goal being to make reproducibility too easy not to do. It's a diverse and welcoming community with over 200 contributors working collaboratively to share their learning and experiences supported by our community manager.

LC: There is also the UK Reproducibility Network, of which many universities are part who offer training including disciplinary approaches to coding. So, there are a few opportunities out there to go and get started!

LC: If we imagine the future 10 years from now, from a reproducibility perspective, what would you like to see in place?

MOR: Here are some of the things I'd like to see:

- for data, I'd like to see strong data provenance, with data being versioned, metadata and example data made available, to help you understand the nature of the data and its properties
- An ability to reproduce input data from original data as well as outputs
- more standard libraries for accessing and reusing data for either preprocessing or combining data
- all papers come with associated code and data, taking into account sensitivity
- 'one-click' reproducibility using some of the great tools are now becoming available
- for sensitive data, the ability to rerun code a secure environment and confirm it produces the results without having to gain access to the underlying data.
- people move from publishing open and reproducible results to **working reproducibly** in the open from the start; for example following the open protocols' communities who share what they are doing in real time
- libraries and archives have a big place to play, hosting longer-term reference computation environments on which to tie our reproducibility stack to
- people making code **reusable**, not just reproducible, so that we're building better foundations for the next wave of advances in our fields.

LC: Thank you Martin. It sounds like a dream, but I'm sure it's probably going to be very realistic and certainly the way we are heading.

Audience questions

Question 1: Have you come across any instances where you feel that data owners could do more to document some of the code from the datasets they publish?

FR: For administrative data, for example the tax data we're working on, that comes out of an admin system but there is little descriptive information about it. A big part of our project is to try to explain what the data is and what are the characteristics of that data. ONS survey data, collected for the purposes of producing National Statistics, is really good on methodology and sampling but not so strong on characteristics of how a variable was derived.

MOR: Not enough data comes with enough information about it. We have had a great push for open government data, but the quality is not always open; in PDF format or CSV where formatting (indentation, colour etc) don't offer semantically meaning, but I do think that is improving. In my experience, for many datasets there are very rarely derived variables that are clearly explained. It seems like there is a lot of waste in the system, where an understanding of what's been has not been made available.

For a sensitive dataset, it can be really hard to understand what you can do with it if you don't have someone on your project who has already used it. You may not be aware of things like missingness or the real structure of the data.

Question 2. There's a lot of platforms and tools have been mentioned but unable to keep track of which ones are best and what they all do. Is there is a good starting point?

MOR: The Turing Way is a good start.

Question 3. What could funders do to incentivize more researchers to share code? Should it come from the community or should journals mandate it?

FR: Funders could certainly it a requirement, but it's hard to enforce. I think making good coding available is something we should promote to extract the most value from a project

MOR: Ideally we should make a convincing case that it's in our own benefit to work this way and align the overall incentives with what counts as impact and contribution in research. We can learn from some of the Open Access

history; data access has been talked about for quite a while with some concrete actions (FAIR) and open access for articles (Plan S), but it's been community action that has helped progress it, rather than mandates.

Question 3: Obviously researchers use different software and the routines are different. Are there ways to cross fertilize code written, say in R with something written in SPSS?

FR: There are a few things you can do on that to make code generally more readable. I usually work in *Stata* and used to work in *Gauss*, and am trying to learn *R.* If you have code that says X1=X2*100, that's going to be unreadable in any language, but if we have a line saying MinWagePennies = MinWagePounds * 100 that suddenly becomes understandable.

All of the modern languages allow you to have very flexible layout, with subroutines to help modularize things. So with a little bit more effort, like clear commenting, you can make your code more readable.

MOR: Using common approaches across a team for example, routines for data cleaning or imputing missing data can be helpful.

LC: I'd like to thank my two interviewees for their words of wisdom and wish them well in their data endeavours. I'd like to encourage readers from the social sciences who are starting their quant analysis careers to be early adopters for code sharing.

On a closing note, in term of closing note on reproducibility from my own area, at the ONS, we are aiming to encourage code sharing within the Secure Research Service (SRS) for researchers who work on common government research datasets, for example, those exploring new linked administrative data may be doing added-value work that others can fruitfully use within a closed environment.

Resources

Secure Research Service, Office for National Statistics (ONS)

- Accessing secure research data as an accredited researcher
- SRS Data Catalogue

Professor Felix Ritchie

- His University of West England webpage
- Wage & Employment Dynamics project

Dr Martin O'Reilly

- His Turing Institute webpage
- The Turing Way online book: https://the-turing-way.netlify.app/
- Get involved with the community: https://github.com/alan-turing-institute/the-turing-way (scroll down to README)

Code collaboration and version control

- GitHub (https://github.com/). Has free tiers for open projects
- GitLab (https://gitlab.com/) - good alternative that also offers free tiers for open projects and a free community edition for running your own instance

- Turing Way chapters on Git (https://the-turing-way.netlify.app/reproducible-research/vcs.html) and GitHub (https://the-turing-way.netlify.app/collaboration/github-novice.html)

Reproducible computational environments

- Python virtual environments (per project packages and versions): https://docs.python-guide.org/dev/virtualenvs/
- R environments (per project packages and versions): https://rstudio.github.io/renv
- Docker (full environments from operating system upwards): https://docs.docker.com/get-started/
- One-click reproducible deployment from a GitHub repository with Binder: https://mybinder.org/

Literate programming (code + results + analysis narrative together)

- Jupyter notebooks: https://jupyter.org/ (supports Python, R, Julia languages + others)
- RMarkdown in RStudio: https://rmarkdown.rstudio.com/

Capturing on-computational elements of research process:

- Open Lab Notebooks: https://openlabnotebooks.org/
- Protocols.io: https://www.protocols.io/

Checking reproducibility / letting others easily run your code

- CodeCheck: https://codecheck.org.uk/
- E-Life Executable Research Articles: https://elifesciences.org/collections/d72819a9/executable-research-articles
- Binder: https://mybinder.org/
- Code Ocean: https://codeocean.com/
- Elsevier Pilot: https://codeocean.com/publisher-signup/elsevier/
- Taylor and Francis pilot: https://authorservices.taylorandfrancis.com/data-sharing/share-your-data/share-code/
- CODECHECK: https://codecheck.org.uk/ offers gold standard checks on code